

OpenGL[®]

Programming Guide

Eighth Edition

*The Official Guide to Learning
OpenGL[®], Version 4.3*



Dave Shreiner • Graham Sellers • John Kessenich • Bill Licea-Kane

The Khronos OpenGL ARB Working Group

Praise for OpenGL[®] Programming Guide, Eighth Edition

“Wow! This book is basically one-stop shopping for OpenGL information. It is the kind of book that I will be reaching for a lot. Thanks to Dave, Graham, John, and Bill for an amazing effort.”

—Mike Bailey, professor, Oregon State University

“The most recent Red Book parallels the grand tradition of OpenGL; continuous evolution towards ever-greater power and efficiency. The eighth edition contains up-to-the minute information about the latest standard and new features, along with a solid grounding in modern OpenGL techniques that will work anywhere. The Red Book continues to be an essential reference for all new employees at my simulation company. What else can be said about this essential guide? I laughed, I cried, it was much better than *Cats*—I’ll read it again and again.”

—Bob Kuehne, president, Blue Newt Software

“OpenGL has undergone enormous changes since its inception twenty years ago. This new edition is your practical guide to using the OpenGL of today. Modern OpenGL is centered on the use of shaders, and this edition of the Programming Guide jumps right in, with shaders covered in depth in Chapter 2. It continues in later chapters with even more specifics on everything from texturing to compute shaders. No matter how well you know it or how long you’ve been doing it, if you are going to write an OpenGL program, you want to have a copy of the OpenGL[®] Programming Guide handy.”

—Marc Olano, associate professor, UMBC

“If you are looking for the definitive guide to programming with the very latest version of OpenGL, look no further. The authors of this book have been deeply involved in the creation of OpenGL 4.3, and everything you need to know about the cutting edge of this industry-leading API is laid out here in a clear, logical, and insightful manner.”

—Neil Trevett, president, Khronos Group

This page intentionally left blank

OpenGL[®]

Programming Guide

Eighth Edition

OpenGL® Series



Visit informit.com/opengl for a complete list of available products

The OpenGL graphics system is a software interface to graphics hardware. (“GL” stands for “Graphics Library.”) It allows you to create interactive programs that produce color images of moving, three-dimensional objects. With OpenGL, you can control computer-graphics technology to produce realistic pictures, or ones that depart from reality in imaginative ways.

The **OpenGL Series** from Addison-Wesley Professional comprises tutorial and reference books that help programmers gain a practical understanding of OpenGL standards, along with the insight needed to unlock OpenGL’s full potential.

PEARSON

◆ Addison-Wesley Cisco Press EXAMCRAM IBM Press. QUE PRENTICE HALL SAMS | Safari

OpenGL[®] Programming Guide Eighth Edition

*The Official Guide to
Learning OpenGL[®], Version 4.3*

*Dave Shreiner
Graham Sellers
John Kessenich
Bill Licea-Kane*

The Khronos OpenGL ARB Working Group

◆◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

OpenGL programming guide : the official guide to learning OpenGL, version 4.3 /
Dave Shreiner, Graham Sellers, John Kessenich, Bill Licea-Kane ; the Khronos OpenGL
ARB Working Group.—Eighth edition.

pages cm

Includes index.

ISBN 978-0-321-77303-6 (pbk. : alk. paper)

1. Computer graphics. 2. OpenGL. I. Shreiner, Dave. II. Sellers, Graham.

III. Kessenich, John M. IV. Licea-Kane, Bill. V. Khronos OpenGL ARB Working Group.

T385.O635 2013

006.6'63—dc23

2012043324

Copyright © 2013 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-77303-6

ISBN-10: 0-321-77303-9

Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.

First printing, March 2013

*For my family—Vicki, Bonnie, Bob, Cookie, Goatee, Phantom, Squiggles,
Tuxedo, and Toby.*
—DRS

To Emily: welcome, we're so glad you're here! Chris and J.: you still rock!
—GJAS

In memory of Phil Karlton, Celeste Fowler, Joan Eslinger, and Ben Cheatham.

This page intentionally left blank

Contents

Figures	xxiii
Tables	xxix
Examples	xxxiii
About This Guide	xli
What This Guide Contains	xli
What's New in This Edition	xliii
What You Should Know Before Reading This Guide	xliii
How to Obtain the Sample Code	xliv
Errata	xlv
Style Conventions	xlvi
1. Introduction to OpenGL	1
What Is OpenGL?	2
Your First Look at an OpenGL Program	3
OpenGL Syntax	8
OpenGL's Rendering Pipeline	10
Preparing to Send Data to OpenGL	11
Sending Data to OpenGL	11
Vertex Shading	12
Tessellation Shading	12
Geometry Shading	12
Primitive Assembly	12
Clipping	13
Rasterization	13
Fragment Shading	13

Per-Fragment Operations	13
Our First Program: A Detailed Discussion	14
Entering main()	14
OpenGL Initialization	16
Our First OpenGL Rendering	28
2. Shader Fundamentals	33
Shaders and OpenGL	34
OpenGL's Programmable Pipeline	35
An Overview of the OpenGL Shading Language	37
Creating Shaders with GLSL	37
Storage Qualifiers	45
Statements	49
Computational Invariance	54
Shader Preprocessor	56
Compiler Control	58
Global Shader-Compilation Option	59
Interface Blocks	60
Uniform Blocks	61
Specifying Uniform Blocks in Shaders	61
Accessing Uniform Blocks from Your Application	63
Buffer Blocks	69
In/Out Blocks	70
Compiling Shaders	70
Our LoadShaders() Function	76
Shader Subroutines	76
GLSL Subroutine Setup	77
Selecting Shader Subroutines	78
Separate Shader Objects	81
3. Drawing with OpenGL	85
OpenGL Graphics Primitives	86
Points	87
Lines, Strips, and Loops	88
Triangles, Strips, and Fans	89
Data in OpenGL Buffers	92
Creating and Allocating Buffers	92
Getting Data into and out of Buffers	95

Accessing the Content of Buffers	100
Discarding Buffer Data	107
Vertex Specification	108
VertexAttribPointer in Depth	108
Static Vertex-Attribute Specification	112
OpenGL Drawing Commands	115
Restarting Primitives	124
Instanced Rendering	128
Instanced Vertex Attributes	129
Using the Instance Counter in Shaders	136
Instancing Redux	139
4. Color, Pixels, and Framebuffers	141
Basic Color Theory	142
Buffers and Their Uses	144
Clearing Buffers	146
Masking Buffers	147
Color and OpenGL	148
Color Representation and OpenGL	149
Vertex Colors	150
Rasterization	153
Multisampling	153
Sample Shading	155
Testing and Operating on Fragments	156
Scissor Test	157
Multisample Fragment Operations	158
Stencil Test	159
Stencil Examples	161
Depth Test	163
Blending	166
Blending Factors	167
Controlling Blending Factors	167
The Blending Equation	170
Dithering	171
Logical Operations	171
Occlusion Query	173
Conditional Rendering	176
Per-Primitive Antialiasing	178

Antialiasing Lines	179
Antialiasing Polygons	180
Framebuffer Objects	180
Renderbuffers	183
Creating Renderbuffer Storage	185
Framebuffer Attachments	187
Framebuffer Completeness	190
Invalidating Framebuffers	192
Writing to Multiple Renderbuffers Simultaneously	193
Selecting Color Buffers for Writing and Reading	195
Dual-Source Blending	198
Reading and Copying Pixel Data	200
Copying Pixel Rectangles	203
5. Viewing Transformations, Clipping, and Feedback	205
Viewing	206
Viewing Model	207
Camera Model	207
Orthographic Viewing Model	212
User Transformations	212
Matrix Multiply Refresher	214
Homogeneous Coordinates	215
Linear Transformations and Matrices	219
Transforming Normals	231
OpenGL Matrices	232
OpenGL Transformations	236
Advanced: User Clipping	238
Transform Feedback	239
Transform Feedback Objects	239
Transform Feedback Buffers	241
Configuring Transform Feedback Varyings	244
Starting and Stopping Transform Feedback	250
Transform Feedback Example—Particle System	252
6. Textures	259
Texture Mapping	261
Basic Texture Types	262
Creating and Initializing Textures	263

Texture Formats	270
Proxy Textures	276
Specifying Texture Data	277
Explicitly Setting Texture Data	277
Using Pixel Unpack Buffers	280
Copying Data from the Framebuffer	281
Loading Images from Files	282
Retrieving Texture Data	287
Texture Data Layout	288
Sampler Objects	292
Sampler Parameters	294
Using Textures	295
Texture Coordinates	298
Arranging Texture Data	302
Using Multiple Textures	303
Complex Texture Types	306
3D Textures	307
Array Textures	309
Cube-Map Textures	309
Shadow Samplers	317
Depth-Stencil Textures	318
Buffer Textures	319
Texture Views	321
Compressed Textures	326
Filtering	329
Linear Filtering	330
Using and Generating Mipmaps	333
Calculating the Mipmap Level	338
Mipmap Level-of-Detail Control	339
Advanced Texture Lookup Functions	340
Explicit Level of Detail	340
Explicit Gradient Specification	340
Texture Fetch with Offsets	341
Projective Texturing	342
Texture Queries in Shaders	343
Gathering Texels	345
Combining Special Functions	345
Point Sprites	346

Textured Point Sprites	347
Controlling the Appearance of Points	350
Rendering to Texture Maps	351
Discarding Rendered Data	354
Chapter Summary	356
Texture Redux	356
Texture Best Practices	357
7. Light and Shadow	359
Lighting Introduction	360
Classic Lighting Model	361
Fragment Shaders for Different Light Styles	362
Moving Calculations to the Vertex Shader	373
Multiple Lights and Materials	376
Lighting Coordinate Systems	383
Limitations of the Classic Lighting Model	383
Advanced Lighting Models	384
Hemisphere Lighting	384
Image-Based Lighting	389
Lighting with Spherical Harmonics	395
Shadow Mapping	400
Creating a Shadow Map	401
8. Procedural Texturing	411
Procedural Texturing	412
Regular Patterns	414
Toy Ball	422
Lattice	431
Procedural Shading Summary	432
Bump Mapping	433
Application Setup	436
Vertex Shader	438
Fragment Shader	439
Normal Maps	441
Antialiasing Procedural Textures	442
Sources of Aliasing	442
Avoiding Aliasing	444

Increasing Resolution	445
Antialiasing High Frequencies	447
Frequency Clamping	457
Procedural Antialiasing Summary	459
Noise	460
Definition of Noise	461
Noise Textures	468
Trade-offs	471
A Simple Noise Shader	472
Turbulence	475
Marble	477
Granite	478
Wood	478
Noise Summary	483
Further Information	483
9. Tessellation Shaders	485
Tessellation Shaders	486
Tessellation Patches	487
Tessellation Control Shaders	488
Generating Output-Patch Vertices	489
Tessellation Control Shader Variables	490
Controlling Tessellation	491
Tessellation Evaluation Shaders	496
Specifying the Primitive Generation Domain	497
Specifying the Face Winding for Generated Primitives	497
Specifying the Spacing of Tessellation Coordinates	498
Additional Tessellation Evaluation Shader layout Options	498
Specifying a Vertex's Position	498
Tessellation Evaluation Shader Variables	499
A Tessellation Example: The Teapot	500
Processing Patch Input Vertices	501
Evaluating Tessellation Coordinates for the Teapot	501
Additional Tessellation Techniques	504
View-Dependent Tessellation	504
Shared Tessellated Edges and Cracking	506
Displacement Mapping	507

10. Geometry Shaders	509
Creating a Geometry Shader	510
Geometry Shader Inputs and Outputs	514
Geometry Shader Inputs	514
Special Geometry Shader Primitives	517
Geometry Shader Outputs	523
Producing Primitives	525
Culling Geometry	525
Geometry Amplification	527
Advanced Transform Feedback	532
Multiple Output Streams	533
Primitive Queries	537
Using Transform Feedback Results	539
Geometry Shader Instancing	549
Multiple Viewports and Layered Rendering	550
Viewport Index	550
Layered Rendering	556
Chapter Summary	559
Geometry Shader Redux	560
Geometry Shader Best Practices	561
11. Memory	563
Using Textures for Generic Data Storage	564
Binding Textures to Image Units	569
Reading from and Writing to Images	572
Shader Storage Buffer Objects	576
Writing Structured Data	577
Atomic Operations and Synchronization	578
Atomic Operations on Images	578
Atomic Operations on Buffers	587
Sync Objects	589
Image Qualifiers and Barriers	593
High Performance Atomic Counters	605
Example	609
Order-Independent Transparency	609

12. Compute Shaders	623
Overview	624
Workgroups and Dispatch	625
Knowing Where You Are	630
Communication and Synchronization	632
Communication	633
Synchronization	634
Examples	636
Physical Simulation	636
Image Processing	642
Chapter Summary	647
Compute Shader Redux	647
Compute Shader Best Practices	648
A. Basics of GLUT: The OpenGL Utility Toolkit	651
Initializing and Creating a Window	652
Accessing Functions	654
Handling Window and Input Events	655
Managing a Background Process	658
Running the Program	658
B. OpenGL ES and WebGL	659
OpenGL ES	660
WebGL	662
Setting up WebGL within an HTML5 page	662
Initializing Shaders in WebGL	664
Initializing Vertex Data in WebGL	667
Using Texture Maps in WebGL	668
C. Built-in GLSL Variables and Functions	673
Built-in Variables	674
Built-in Variable Declarations	674
Built-in Variable Descriptions	676
Built-in Constants	684
Built-in Functions	686
Angle and Trigonometry Functions	688
Exponential Functions	690
Common Functions	692
Floating-Point Pack and Unpack Functions	698

Geometric Functions	700
Matrix Functions.	702
Vector Relational Functions	703
Integer Functions	705
Texture Functions.	708
Atomic-Counter Functions.	722
Atomic Memory Functions	723
Image Functions	725
Fragment Processing Functions.	729
Noise Functions	731
Geometry Shader Functions	732
Shader Invocation Control Functions	734
Shader Memory Control Functions.	734
D. State Variables	737
The Query Commands.	738
OpenGL State Variables.	745
Current Values and Associated Data.	746
Vertex Array Object State	747
Vertex Array Data	749
Buffer Object State.	750
Transformation State.	751
Coloring State.	752
Rasterization State	753
Multisampling	755
Textures.	756
Textures	759
Textures	762
Textures.	764
Texture Environment.	766
Pixel Operations.	767
Framebuffer Controls	770
Framebuffer State	771
Framebuffer State	772
Framebuffer State.	773
Renderbuffer State	775
Renderbuffer State	776
Pixel State	778

Shader Object State	781
Shader Program Pipeline Object State	782
Shader Program Object State	783
Program Interface State	793
Program Object Resource State	794
Vertex and Geometry Shader State	797
Query Object State	797
Image State	798
Transform Feedback State	799
Atomic Counter State	800
Shader Storage Buffer State	801
Sync Object State	802
Hints	803
Compute Dispatch State	803
Implementation-Dependent Values	804
Tessellation Shader Implementation-Dependent Limits	810
Geometry Shader Implementation-Dependent Limits	813
Fragment Shader Implementation-Dependent Limits	815
Implementation-Dependent Compute Shader Limits	816
Implementation-Dependent Shader Limits	818
Implementation-Dependent Debug Output State	823
Implementation-Dependent Values	824
Internal Format-Dependent Values	826
Implementation-Dependent Transform Feedback Limits	826
Framebuffer-Dependent Values	827
Miscellaneous	827
E. Homogeneous Coordinates and Transformation Matrices	829
Homogeneous Coordinates	830
Transforming Vertices	830
Transforming Normals	831
Transformation Matrices	831
Translation	832
Scaling	832
Rotation	832
Perspective Projection	834
Orthographic Projection	834

F. OpenGL and Window Systems	835
Accessing New OpenGL Functions	836
GLEW: The OpenGL Extension Wrangler	837
GLX: OpenGL Extension for the X Window System	838
Initialization	839
Controlling Rendering	840
GLX Prototypes	842
WGL: OpenGL Extensions for Microsoft Windows	845
Initialization	846
Controlling Rendering	846
WGL Prototypes	848
OpenGL in Mac OS X: The Core OpenGL (CGL) API and the NSOpenGL Classes	850
Mac OS X's Core OpenGL Library	851
Initialization	851
Controlling Rendering	852
CGL Prototypes	852
The NSOpenGL Classes	854
Initialization	854
G. Floating-Point Formats for Textures, Framebuffers, and Renderbuffers	857
Reduced-Precision Floating-Point Values	858
16-bit Floating-Point Values	858
10- and 11-bit Unsigned Floating-Point Values	860
H. Debugging and Profiling OpenGL	865
Creating a Debug Context	866
Debug Output	868
Debug Messages	869
Filtering Messages	872
Application-Generated Messages	874
Debug Groups	875
Naming Objects	877
Profiling	879
Profiling Tools	879
In-Application Profiling	881

I. Buffer Object Layouts	885
Using Standard Layout Qualifiers	886
The <code>std140</code> Layout Rules	886
The <code>std430</code> Layout Rules	887
Glossary	889
Index	919

This page intentionally left blank

Figures

Figure 1.1	Image from our first OpenGL program: triangles.cpp	5
Figure 1.2	The OpenGL pipeline	10
Figure 2.1	Shader-compilation command sequence	71
Figure 3.1	Vertex layout for a triangle strip	89
Figure 3.2	Vertex layout for a triangle fan	90
Figure 3.3	Packing of elements in a BGRA-packed vertex attribute	112
Figure 3.4	Packing of elements in a RGBA-packed vertex attribute	112
Figure 3.5	Simple example of drawing commands	124
Figure 3.6	Using primitive restart to break a triangle strip	125
Figure 3.7	Two triangle strips forming a cube	127
Figure 3.8	Result of rendering with instanced vertex attributes	134
Figure 3.9	Result of instanced rendering using <code>gl_InstanceID</code>	139
Figure 4.1	Region occupied by a pixel	144
Figure 4.2	Polygons and their depth slopes	165
Figure 4.3	Aliased and antialiased lines	178
Figure 4.4	Close-up of RGB color elements in an LCD panel	199
Figure 5.1	Steps in configuring and positioning the viewing frustum	207
Figure 5.2	Coordinate systems required by OpenGL	209
Figure 5.3	User coordinate systems unseen by OpenGL	210
Figure 5.4	A view frustum	211
Figure 5.5	Pipeline subset for user/shader part of transforming coordinates	212

- [click Climate Change in California: Risk and Response](#)
- [download Cracked: The Unhappy Truth about Psychiatry](#)
- [Genki I: An Integrated Course in Elementary Japanese pdf, azw \(kindle\)](#)
- [Nietzsche: A Philosophical Biography for free](#)

- <http://academialanguagebar.com/?ebooks/Climate-Change-in-California--Risk-and-Response.pdf>
- <http://schroff.de/books/Rico-Slade-Will-Fucking-Kill-You.pdf>
- <http://musor.ruspb.info/?library/Genki-I--An-Integrated-Course-in-Elementary-Japanese.pdf>
- <http://patrickvincitore.com/?ebooks/Olive--June-2016-.pdf>